mogsa: gene set analysis on multiple omics data

Chen Meng

Modified: January 19, 2016. Compiled: June 7, 2016.

Contents

1 MOGSA overview

Modern "omics" technologies enable quantitative monitoring of the abundance of various biological molecules in a high-throughput manner, accumulating an unprecedented amount of quantitative information on a genomic scale. Gene set analysis is a particularly useful method in high throughput data analysis since it can summarize single gene level information into the biological informative gene set levels. The *mogsa* provide a method doing gene set analysis based on multiple omics data that describes the same set of observations/samples.

MOGSA algorithm consists of three steps. In the first step, multiple omics data are integrated using multi-table multivariate analysis, such as multiple factorial analysis (MFA) [**?**]. MFA projects the observations and variables (genes) from each dataset onto a lower dimensional space, resulting in sample scores (or PCs) and variables loadings respectively. Next, gene set annotations are projected as additional information onto the same space, generating a set of scores for each gene set across samples [**?**]. In the final step, MOGSA generates a gene set score (GSS) matrix by reconstructing the sample scores and gene set scores. A high GSS indicates that gene set and the variables in that gene set have measurement in one or more dataset that explain a large proportion of the correlated information across data tables. Variables (genes) unique to individual datasets or common among matrices may contribute to a high GSS. For example, in a gene set, a few genes may have high levels of gene expression, others may have increased protein levels and a few may have amplifications in copy number.

In this document, we show with an example how to use MOGSA to integrate and annotate multiple omics data.

2 Run mogsa

2.1 Quick start

In this working example, we will analyze the NCI-60 transcriptomic data from 4 different microarray platforms. The goal is to explore which functions (gene sets) are associated with (high or low expressed) which type of tumor. First, load the library and data

```
# loading gene expression data and supplementary data
library(mogsa)
library(gplots) # used for visulizing heatmap
# loading gene expression data and supplementary data
data(NCI60_4array_supdata)
data(NCI60_4arrays)
```

`NCI60_4arrays` is a *list* of *data.frame*. The *list* consists of microarray data for NCI-60 cell lines from different platforms. In each of the *data.frame*, columns are the 60 cell lines and rows are genes. The data was downloaded from [**?**], but only a small subset of genes were selected. Therefore, the result in this vignette is not intended for biological interpretation.

`NCI60_4array_supdata` is a *list* of *matrix*, representing gene set annotation data. For each of the microarray data, there is a corresponding annotation matrix. In the annotation data, the rows are genes (in the same order as their original dataset) and columns are gene sets. An annotation matrix is a binary matrix, where 1 indicates a gene is present in a gene set and 0 otherwise. See the "Preparation of gene set data" section about how to create the gene set annotation matrices as required by `mogsa`. To have an overview of the two datasets:

```
sapply(NCI60_4arrays, dim) # check dimensions of expression data
```

```
##      agilent hgu133 hgu133p2 hgu95
## [1,]     300    298      268   288
## [2,]      60     60       60    60
```

```
sapply(NCI60_4array_supdata, dim) # check dimensions of supplementary data
```

```
##      agilent hgu133 hgu133p2 hgu95
## [1,]     300    298      268   288
## [2,]     150    150      150   150
```

```
# check if the gene expression data and annotation data are mathced in the same order
identical(names(NCI60_4arrays), names(NCI60_4array_supdata))
```

```
## [1] TRUE
```

```
head(rownames(NCI60_4arrays$agilent)) # the type of gene IDs
```

```
## [1] "ST8SIA1" "YWHAQ"   "EPHA4"   "GTPBP5"  "PVR"     "ATP6V1H"
```

Also, we need to confirm the columns between the expression data and annotation data are mapped in the same order. To verify this, we do

```
dataColNames <- lapply(NCI60_4arrays, colnames)
supColNames <- lapply(NCI60_4arrays, colnames)
identical(dataColNames, supColNames)
```

```
## [1] TRUE
```

Before applying MOGSA, we first define a factor describing the tissue of origin of cell lines and color code, which will be used later.

```
# define cancer type
cancerType <- as.factor(substr(colnames(NCI60_4arrays$agilent), 1, 2))
# define color code to distinguish cancer types
colcode <- cancerType
levels(colcode) <- c("black", "red", "green", "blue",
                     "cyan", "brown", "pink", "gray", "orange")
colcode <- as.character(colcode)
```

Then, we call the function `mogsa` to run MOGSA:

```
mgsa1 <- mogsa(x = NCI60_4arrays, sup=NCI60_4array_supdata, nf=3,
               proc.row = "center_ssq1", w.data = "inertia", statis = TRUE)
```

In this function, the input argument `proc.row` stands for the preprocessing of rows and argument `w.data` indicates the weight of datasets. The last argument `statis` is about which multiple table analysis method should be used. Two multivariate methods are available at present, one is "STATIS" (`statis=TRUE`) [**?**], the other one is multiple factorial analysis (MFA; `statis=FALSE, the default setting`) [**?**].

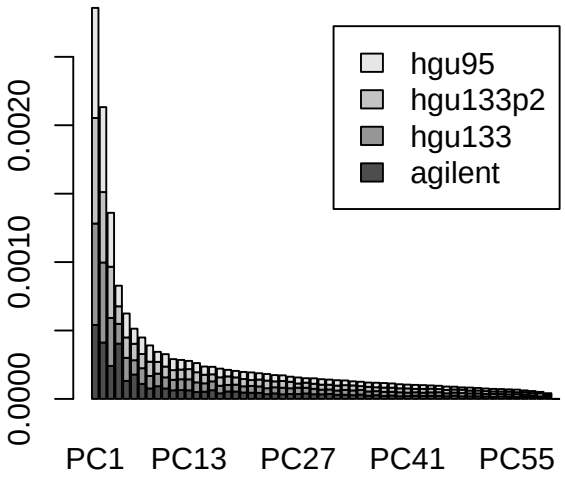

Figure 1: The variance of each principal components (PC), the contributions of different data are distinguished by different colors

In this analysis, we arbitrarily selected top three PCs (nf=3). But in practice, the number of PCs need to be determined before running the MOGSA. Therefore, it is also possible to run the multivariate analysis and projecting annotation data separately. After running the multivariate analysis, a scree plot of eigenvalues for each PC could be used to determine the proper number of PCs to be included in the annotation projection step (See the "Perform MOGSA in two steps" section).

2.2 Result analysis and interpretation

The function `mogsa` returns an object of class *mgsa*. This information could be extracted with function `getmgsa`. First, we want to know the variance explained by each PC on different datasets (figure 1).

```
eigs <- getmgsa(mgsa1, "partial.eig") # get partial "eigenvalue" for separate data
barplot(as.matrix(eigs), legend.text = rownames(eigs))
```

The main result returned by `mogsa` is the gene set score (GSS) matrix. The value in the matrix indicates the overall active level of a gene set in a sample. The matrix could be extracted and visualized by

```
# get the score matrix
scores <- getmgsa(mgsa1, "score")
heatmap.2(scores, trace = "n", scale = "r", Colv = NULL, dendrogram = "row",
          margins = c(6, 10), ColSideColors=colcode)
```

Figure 2 shows the gene set score matrix returned by `mogsa`. The rows of the matrix are all the gene sets used to annotate the data. But we are mostly interested in the gene sets with large number of significant gene sets, because these gene sets describe the difference across cell lines. The corresponding p-value for each gene set score could be extracted by `getmgsa`. Then, the most significant gene sets could be defined as gene sets that contain highest number of significantly p-values. For example, if we want to select the top 20 most significant gene

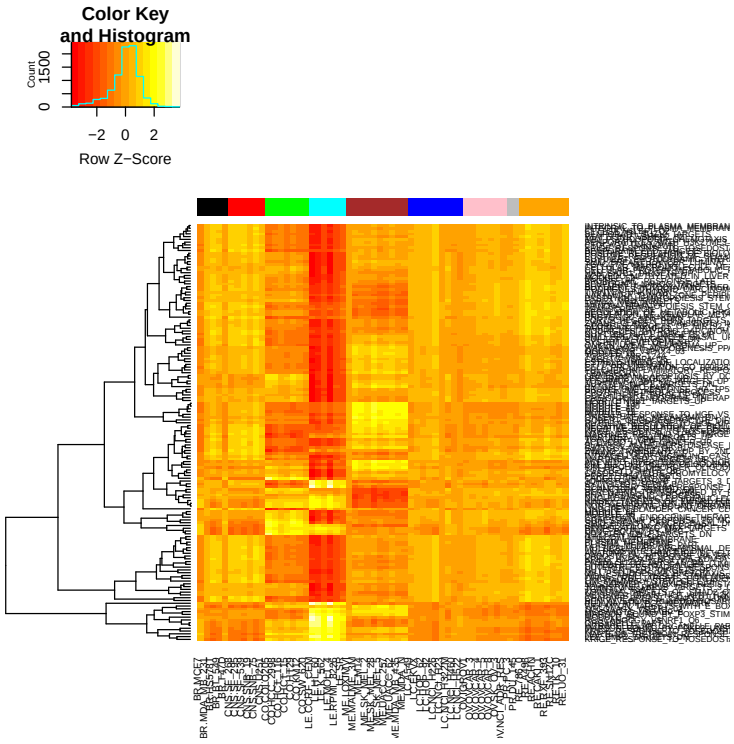

Figure 2: heatmap showing the gene set score (GSS) matrix

sets and plot them in heatmap, we do:

```
p.mat <- getmgsa(mgsa1, "p.val") # get p value matrix
# select gene sets with most signficant GSS scores.
top.gs <- sort(rowSums(p.mat < 0.01), decreasing = TRUE)[1:20]
top.gs.name <- names(top.gs)
top.gs.name
```

```
##  [1] "PASINI_SUZ12_TARGETS_DN"
##  [2] "CHARAFE_BREAST_CANCER_LUMINAL_VS_BASAL_DN"
##  [3] "CHARAFE_BREAST_CANCER_LUMINAL_VS_MESENCHYMAL_DN"
##  [4] "KOINUMA_TARGETS_OF_SMAD2_OR_SMAD3"
##  [5] "DUTERTRE_ESTRADIOL_RESPONSE_24HR_DN"
##  [6] "REN_ALVEOLAR_RHABDOMYOSARCOMA_DN"
##  [7] "LIM_MAMMARY_STEM_CELL_UP"
##  [8] "LIU_PROSTATE_CANCER_DN"
##  [9] "CHICAS_RB1_TARGETS_CONFLUENT"
## [10] "NUYTTEN_EZH2_TARGETS_UP"
## [11] "DACOSTA_UV_RESPONSE_VIA_ERCC3_DN"
## [12] "PUJANA_ATM_PCC_NETWORK"
## [13] "KRIGE_RESPONSE_TO_TOSEDOSTAT_24HR_DN"
## [14] "WONG_ADULT_TISSUE_STEM_MODULE"
## [15] "KRIEG_HYPOXIA_NOT_VIA_KDM3A"
## [16] "MULTICELLULAR_ORGANISMAL_DEVELOPMENT"
## [17] "ANATOMICAL_STRUCTURE_DEVELOPMENT"
## [18] "FORTSCHEGGER_PHF8_TARGETS_DN"
## [19] "ZWANG_CLASS_1_TRANSIENTLY_INDUCED_BY_EGF"
## [20] "PLASMA_MEMBRANE_PART"
```

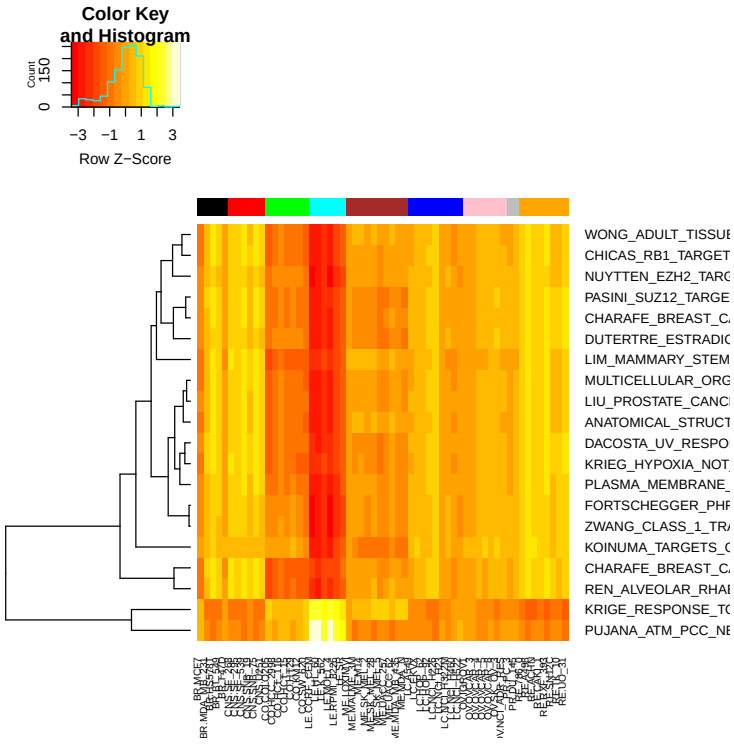

Figure 3: heatmap showing the gene set score (GSS) matrix for top 20 significant gene sets

```
heatmap.2(scores[top.gs.name, ], trace = "n", scale = "r", Colv = NULL, dendrogram = "row",
          margins = c(6, 10), ColSideColors=colcode)
```

The result is shown in figure 3. We can see that these gene sets reflect the difference between leukemia and other tumors.

So far, we already had an integrative overview of gene sets active levels over the 60 cell lines. It is also interesting to look into more detailed information for a specific gene set. For example, which dataset(s) contribute most to the high or low gene set score of a gene set? And which genes are most important in defining the gene set score for a gene set? The former question could be answered by the gene set score decomposition; the later question could be solve by the gene influential score. These analysis can be done with `decompose.gs.group` and `GIS`.

In the first example, we explore the gene set that have most significant gene set scores. The gene set is

```
# gene set score decomposition
# we explore two gene sets, the first one
gs1 <- top.gs.name[1] # select the most significant gene set
gs1
```

```
## [1] "PASINI_SUZ12_TARGETS_DN"
```

The data-wise decomposition of this gene set over cancer types is

```
# decompose the gene set score over datasets
decompose.gs.group(mgsa1, gs1, group = cancerType)
```

Figure 4 shows leukemia cell lines have lowest GSS on this gene set. The contribution to the overall gene set score by each dataset are separated in this plot. In general, there is a good concordance between different datasets. But HGU133 platform contribute most and Agilent platform contributed least comparing with other datasets, represented as the longest or shortest bars.

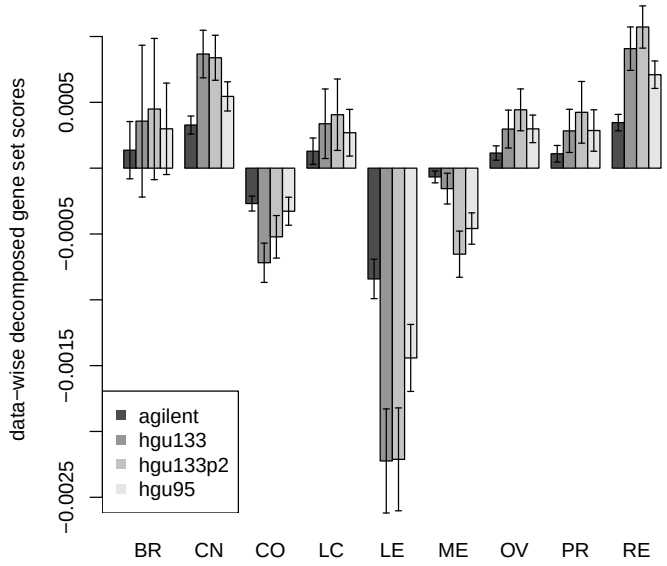

Figure 4: gene set score (GSS) decomposition. The GSS decomposition are grouped according to the tissue of origin of cell lines. The vertical bar showing the 95% of confidence interval of the means.

Next, in order to know the most influential genes in this gene set. We call the function `GIS`:

```
gis1 <- GIS(mgsa1, gs1, barcol = gray.colors(4)) # gene influential score
```

```
head(gis1) # print top 6 influencers
```

```
##   feature      GIS    data
## 1   LIMD2 1.007091  hgu133
## 2  ZNF266 1.006706  hgu133
## 3   LIMD2 1.006476   hgu95
## 4    GNG2 1.006327 agilent
## 5     SP5 1.006035   hgu95
## 6     SP5 1.005954  hgu133
```

In figure 5, the bars represent the gene influential scores for genes. Genes from different platforms are shown in different colors. The expression of genes with high positive GIS more likely to have a good positive correlation with the gene set score. In this example, the most important genes in the gene set ”PASIN SUZ12 TARGETS DN” are TNFRSF12A (identified in two different platforms), CD151, ITGB1, etc.

In the next example, we use the same methods to explore the ”PUJANA ATM PCC NETWORK” gene set.

```
# the section gene set
gs2 <- "PUJANA_ATM_PCC_NETWORK"
decompose.gs.group(mgsa1, gs2, group = cancerType, x.legend = "topright")
```

```
gis2 <- GIS(mgsa1, "PUJANA_ATM_PCC_NETWORK", topN = 6, barcol = gray.colors(4))
```

```
gis2
```

```
##   feature      GIS     data
```

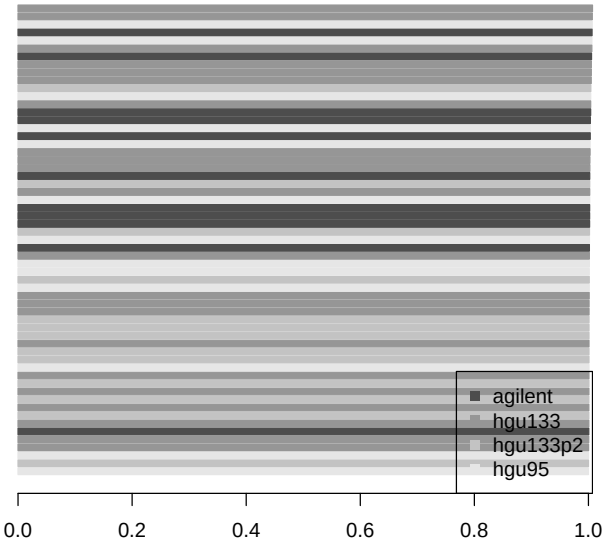

Figure 5: The gene influential score (GIS) plot. the GIS are represented as bars and the original data where the gene is from is distingished by different colors.

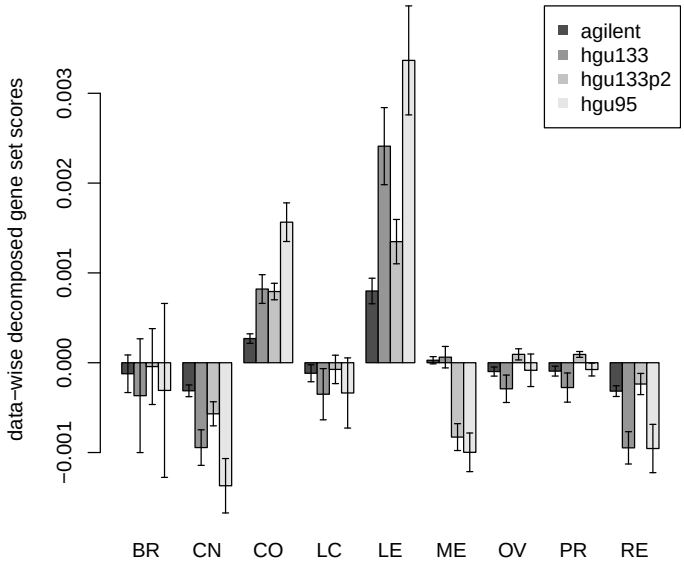

Figure 6: Data-wise decomposed GSS for gene set 'PUJANA ATM PCC NETWORK'

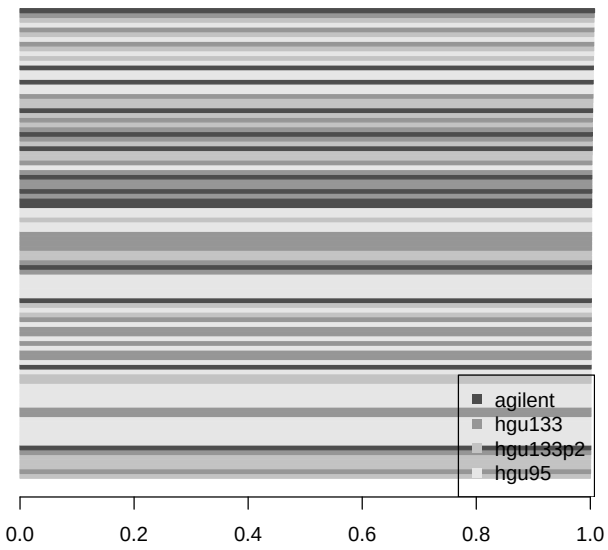

Figure 7: GIS plot for gene set 'PUJANA ATM PCC NETWORK'

```
##   PCBP4 1.007281  agilent
##     LIF 1.006737   hgu133
##    DKK3 1.006393 hgu133p2
##   ROBO1 1.006231    hgu95
##    GPD2 1.006213   hgu133
##  KCNMA1 1.006116 hgu133p2
```

Figure 6 shows that the the leukemia cell lines have highest GSSs for this gene set. And the HGU133 and HGU95 platform have relative high contribution to the overall gene set score. The GIS analysis (figure 7) indicates the PIK4CG and GMFG are the most important genes in this gene set.

2.3 Plot gene sets in projected space

We can also see how the gene set are presented in the lower dimension space. Here we show the projection of gene set annotations on first two dimensions. Then, the label the two gene sets we analyzed before.

```
fs <- getmgsa(mgsa1, "fac.scr") # extract the factor scores for cell lines (cell line space)
layout(matrix(1:2, 1, 2))
plot(fs[, 1:2], pch=20, col=colcode, axes = FALSE)
abline(v=0, h=0)
legend("topright", col=unique(colcode), pch=20, legend=unique(cancerType), bty = "n")
plotGS(mgsa1, label.cex = 0.8, center.only = TRUE, topN = 0, label = c(gs1, gs2))
```

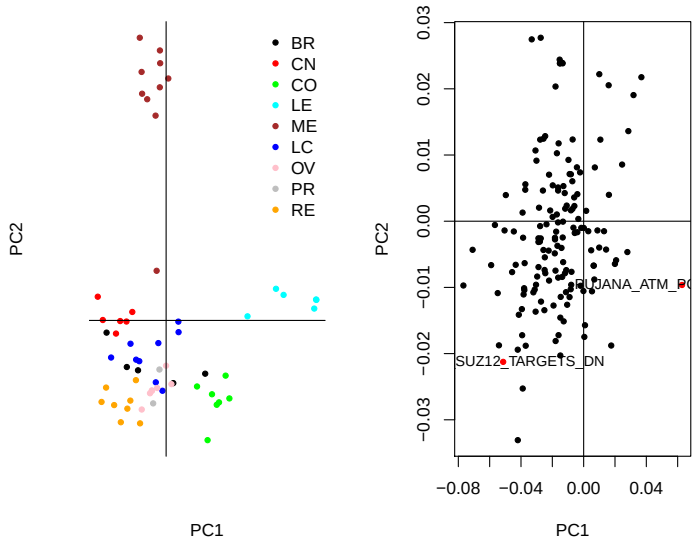

Figure 8: cell line and gene sets projected on the PC1 and PC2

2.4 Perform MOGSA in two steps

`mogsa` perform MOGSA in one step. But in practice, one need to determine how many PCs should be retained in the step of reconstructing gene set score matrix. A scree plot of the eigenvalues, which result from the multivariate analysis, could be used for this purpose. Therefore, we can perform the multivariate data analysis and gene set annotation projection in two steps. To do the multivariate analysis, we call the `moa`:

```
# perform multivariate analysis
ana <- moa(NCI60_4arrays, proc.row = "center_ssq1", w.data = "inertia", statis = TRUE)
slot(ana, "partial.eig")[, 1:6] # extract the eigenvalue
```

```
##                    PC1           PC2           PC3           PC4           PC5           PC6
## agilent  0.0005406833 0.0004119778 0.0002410063 0.0004038087 0.0001317894 0.0001783712
## hgu133   0.0007410830 0.0005850680 0.0003507538 0.0001448788 0.0001685482 0.0001042850
## hgu133p2 0.0007716595 0.0005146566 0.0003742008 0.0001281515 0.0001487516 0.0001203610
## hgu95    0.0008042677 0.0006210049 0.0003942394 0.0001506287 0.0001752495 0.0001102364
```

```
# show the eigenvalues in scree plot:
layout(matrix(1:2, 1, 2))
plot(ana, value="eig", type = 2, n=20, main="variance of PCs") # use '?"moa-class"' to check
plot(ana, value="tau", type = 2, n=20, main="Scaled variance of PCs")
```

The multivariate analysis (`moa`) returns an object of class *moa-class*. The scree plot shows the top 3 PC is the most significant since they explain much more variance than others. Several other methods, such as the informal "elbow test" or more formal test could be used to determine the number of retained PCs [**?**]. In order to be consistent with previous example, we use top 3 PCs in the analysis:

```
mgsa2 <- mogsa(x = ana, sup=NCI60_4array_supdata, nf=3)
```

```
## x is an object of "moa", statis is not used
```

```
identical(mgsa1, mgsa2) # check if the two methods give the same results
```

```
## [1] FALSE
```

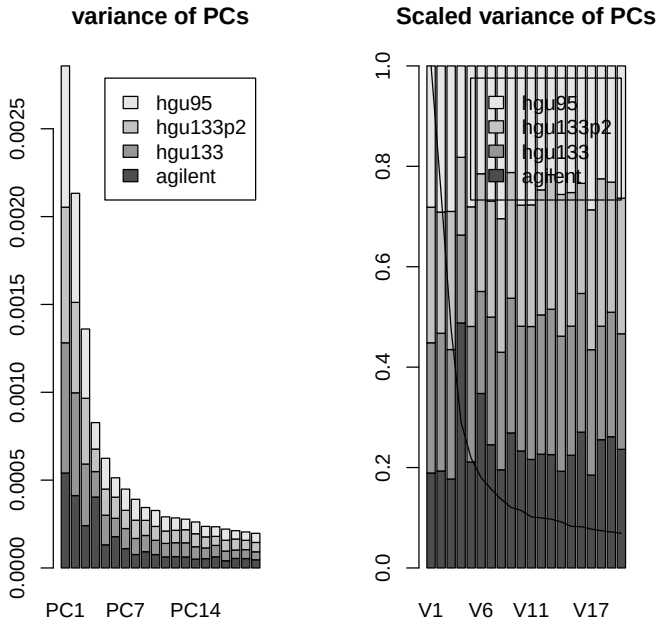

Figure 9: cell line and gene sets projected on the PC1 and PC2

3 Preparation of gene set data

Package *GSEABase* provides several methods to create a gene set list [**?**]. In *mogsa* there are two methods to create gene set list. The first one is generating gene set list from package *graphite* [**?**] using function `prepGraphite`.

```
library(graphite)
keggdb <- prepGraphite(db = pathways("hsapiens", "kegg")[1:50], id = "symbol")

## converting identifiers!
## converting identifiers done!

keggdb[1:2]

## $`Acute myeloid leukemia`
##  [1] "AKT3"     "CHUK"     "AKT1"     "AKT2"     "FLT3"     "PIK3R5"   "MTOR"
##  [8] "GRB2"     "HRAS"     "IKBKB"    "ARAF"     "JUP"      "KIT"      "KRAS"
## [15] "NRAS"     "PIK3CA"   "PIK3CB"   "PIK3CD"   "PIK3CG"   "PIK3R1"   "PIK3R2"
## [22] "PML"      "MAP2K1"   "MAP2K2"   "RAF1"     "RARA"     "SOS1"     "SOS2"
## [29] "BRAF"     "STAT3"    "STAT5A"   "STAT5B"   "ZBTB16"   "PIK3R3"   "IKBKG"
## [36] "RUNX1"    "RUNX1T1"  "BAD"      "NFKB1"    "RELA"     "CEBPA"    "SPI1"
## [43] "EIF4EBP1" "RPS6KB1"  "RPS6KB2"  "MYC"      "LEF1"     "PPARD"    "CCND1"
## [50] "TCF7"     "TCF7L2"   "TCF7L1"   "CCNA1"    "MAPK1"    "MAPK3"    "PIM2"
## [57] "PIM1"
##
## $`Adherens junction`
##  [1] "WASF2"    "BAIAP2"   "SORBS1"   "WASF3"    "SSX2IP"   "CSNK2A1" "CSNK2A2" "CSNK2B"
##  [9] "CTNNA1"   "CTNNA2"   "CTNND1"   "EGFR"     "ERBB2"    "FER"      "FGFR1"    "FYN"
## [17] "NECTIN3" "CSNK2A3" "CTNNA3"   "IGF1R"    "RHOA"     "LMO7"     "SMAD2"    "SMAD3"
```

```
## [25] "MET"     "MLLT4"   "LEF1"    "ACP1"    "MAPK1"   "MAPK3"   "PTPN1"   "PTPN6"
## [33] "PTPRB"   "PTPRF"   "PTPRJ"   "PTPRM"   "NECTIN1" "NECTIN2" "RAC1"    "RAC2"
## [41] "RAC3"    "ACTB"    "SRC"     "MAP3K7"  "TCF7"    "TCF7L2"  "TGFBR1"  "TGFBR2"
## [49] "ACTG1"   "VCL"     "WAS"     "YES1"    "ACTN4"   "NECTIN4" "TCF7L1"  "ACTN1"
## [57] "ACTN2"   "IQGAP1"  "ACTN3"   "WASF1"   "WASL"    "FARP2"   "CDC42"   "CDH1"
## [65] "CTNNB1"  "TJP1"    "SNAI1"   "PARD3"   "SMAD4"   "SNAI2"   "NLK"
```

The second method is to create a gene set list from "gmt" files, which could be downloaded from MSigDB [**?**] after obtaining a proper license. In our working example, we will work on a toy example from this database containing only three datasets.

```
dir <- system.file(package = "mogsa")
preGS <- prepMsigDB(file=paste(dir, "/extdata/example_msigdb_data.gmt.gz", sep = ""))
```

In order to use the gene set information in `mogsa`, we have to convert the list of gene sets to a list of annotation matrix. This can be done with `prepSupMoa`. This function requires two obligatory inputs, first is the multiple omics datasets and the second input could be a gene set list, *GeneSet* or *GeneSetCollection*. The output of `prepSupMoa` could be directly passed into the `mogsa`.

```
# the prepare
sup_data1 <- prepSupMoa(NCI60_4arrays, geneSets=keggdb)
mgsa3 <- mogsa(x = NCI60_4arrays, sup=sup_data1, nf=3,
               proc.row = "center_ssq1", w.data = "inertia", statis = TRUE)
```

4 Session info

```
toLatex(sessionInfo())
```

- R version 3.3.0 (2016-05-03), `x86_64-apple-darwin13.4.0`
- Locale: `C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8`
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Other packages: knitr 1.13, mogsa 1.6.4
- Loaded via a namespace (and not attached): AnnotationDbi 1.34.3, Biobase 2.32.0, BiocGenerics 0.18.0, BiocStyle 2.0.2, DBI 0.4-1, GSEABase 1.34.0, IRanges 2.6.0, KernSmooth 2.23-15, Matrix 1.2-6, RSQLite 1.0.0, S4Vectors 0.10.1, XML 3.98-1.4, annotate 1.50.0, bitops 1.0-6, caTools 1.17.1, cluster 2.0.4, codetools 0.2-14, corpcor 1.6.8, digest 0.6.9, evaluate 0.9, formatR 1.4, gdata 2.17.0, genefilter 1.54.2, gplots 3.0.1, graph 1.50.0, graphite 1.18.0, grid 3.3.0, gtools 3.5.0, highr 0.6, lattice 0.20-33, magrittr 1.5, parallel 3.3.0, rappdirs 0.3.1, splines 3.3.0, stats4 3.3.0, stringi 1.1.1, stringr 1.0.0, survival 2.39-4, svd 0.4, tools 3.3.0, xtable 1.8-2